In The Claims:

1. An apparatus for pseudo-random testing binary emulation, comprising:

a random code generator that generates an initial machine state and a binary instruction sequence;

a native architecture execution engine that executes the binary instruction sequence to produce a final state S1;

a target architecture execution engine that executes the binary instruction sequence to produce a final state S2, the target architecture execution engine comprising a binary emulator that emulates the binary instruction sequence according to the target architecture; and

a verification engine that compares the final state S1 and the final state S2, wherein when the final state S1 and the final state S2 do not match, an emulation failure has occurred.

2. The apparatus of claim 1, wherein the native and the target architecture execution engines communicate using machine-to-machine communications protocols.

3. The apparatus of claim 2, wherein the communications protocol is a TCP/IP protocol.

4. The apparatus of claim 1, wherein when the emulation failure occurs, information related to the emulation failure is written to a file.

5. The apparatus of claim 1, wherein the target platform is a simulator.

6. The apparatus of claim 1, wherein the target platform is a hardware embodiment.

7. The apparatus of claim 1, wherein the random code generator comprises a probability file comprising probability values for each instruction in a native instructions set architecture (ISA).

8. The apparatus of claim 7, wherein the probability values in the probability file are user-generated.

9. A method for pseudo-random testing of binary emulation, comprising:

generating a pseudo-random binary instruction sequence;

generating an initial machine state;

initializing a native machine according to the initial machine state;

executing the binary instruction sequence on the native machine to produce a final state S1;

initializing a target machine according to the initial machine state;

emulating the binary instruction sequence on the target machine;

executing the emulated binary instruction sequence to produce a final state S2; and

comparing the final state S1 to the final state S2 to determine an emulation error.

10. The method of claim 9, further comprising:

communicating the initial machine state and the binary instruction sequence to the target machine using a machine-to-machine communication protocol; and

communicating the final state S2 to the native machine using the machine-to-machine communication protocol.

11. The method of claim 10, wherein the machine-to-machine communication protocol is a TCP/IP protocol.

12. The method of claim 9, further comprising storing information related to the emulation failure, the information comprising the initial machine state, the binary instruction sequence and the final states S1 and S2.

13. The method of claim 9, wherein the binary emulation is executed on a simulator.

14. The method of claim 9, wherein the binary emulation is executed on a hardware device.

15. The method of claim 9, further comprising:

assigning a probability value to each binary instruction in a native instruction set architecture (ISA);

pseudo-randomly generating a probability value; and

selecting the binary instruction sequence based on the probability value.

16. A method for verifying cross-architecture emulation, comprising:

randomly generating a native instruction sequence and an initial machine state;

providing the native instruction sequence and the initial machine state to a first platform having a first instruction set architecture (ISA);

providing the native instruction sequence to a second platform having a second ISA;

emulating the native instruction sequence according to the second ISA;

executing the native instruction sequence in the first platform, the execution providing a final state S1;

executing the emulated native instruction sequence on the second platform, the execution providing a final state S2; and

comparing the final states S1 and S2 to determine an emulation failure.

17. The method of claim 16, wherein the native instruction sequence is randomly generated from a set of all instructions in the first ISA.

18. The method of claim 16, wherein the random generation is based on a user-defined value assigned to each instruction in the first ISA.

19. The method of claim 16, wherein the emulation of the native instruction sequence and the execution of the emulated native instruction sequence is completed on a binary instruction emulator operating on a simulator which operates on the first platform.

20. The method of claim 16, wherein the emulation is completed on a binary instruction emulator operating on the second platform.